Practical Privacy-Preserving Indoor Localization based on Secure Two-Party Computation

Nieminen, Raine

2021-09-01

Nieminen , R & Järvinen , K 2021 , ' Practical Privacy-Preserving Indoor Localization based on Secure Two-Party Computation ' , IEEE Transactions on Mobile Computing , vol. 20 , no. 9 , 9079655 , pp. 2877-2890 . https://doi.org/10.1109/TMC.2020.2990871

http://hdl.handle.net/10138/333295
https://doi.org/10.1109/TMC.2020.2990871

acceptedVersion

Practical Privacy-Preserving Indoor Localization based on Secure Two-Party Computation

Raine Nieminen and Kimmo Järvinen

Abstract —We present a privacy-preserving indoor localization scheme based on received signal strength measurements, e.g., from WiFi access points. Our scheme preserves the privacy of both the client's location and the service provider's database by using secure two-party computation instantiated with known cryptographic primitives, namely, Paillier encryption and garbled circuits. We describe a number of optimizations that reduce the computation and communication overheads of the scheme and provide theoretical evaluations of these overheads. We also demonstrate the feasibility of the scheme by developing a proof-of-concept implementation for Android smartphones and commodity servers. This implementation allows us to validate the practical performance of our scheme and to show that it is feasible for practical use in certain types of indoor localization applications.

Index Terms —Indoor localization, location privacy, WiFi ngerprinting, secure multi-party computation, Paillier encryption, garbled circuits, Android smartphones

F

1 Introduction

With the rise of smartphones and other smart mobile devices, location data has become an important asset that is used in various Location-Based Services (LBSs) ranging from traditional navigation and map applications to social media and targeted advertising. An obvious prerequisite for any LBS is localization, the process of obtaining the physical location of a client (or more precisely the client's device). In outdoor environments localization is predominately based on Global Navigation Satellite Systems (GNSSs) such as GPS or Galileo. However, GNSS satellite signals are very weak and effectively blocked by physical obstacles leading to poor localization service particularly in indoor environments but even in certain outdoor environments (e.g., dense woods or urban canyons). Hence, alternative localization techniques are required for providing LBSs for indoor environments.

Providing accurate indoor localization is important in order to facilitate useful LBSs such as indoor navigation, e.g., for shopping malls, airports, exhibition centers, hospitals, university campuses, etc. [1], [2], [3]. For instance, a navigation application for a shopping mall will help customers to nd stores easily; retailers naturally bene t from this also, and they may get further bene ts through targeted location-based advertising. To answer the need for indoor localization, many indoor localization techniques have been proposed in the literature and also deployed in practice (see, [4], [5] for surveys) based on using WiFi [6], [7], [8], [9], [10], cellular [11], RFID [12], Bluetooth [13], or Zigbee [14] signals. Localization based on WiFi Received Signal Strength (RSS) ngerprinting is particularly tempting because (a) many indoor areas already have an extensive WiFi infrastructure readily installed leading to small cost of deployment for the Service Provider (SP) and (b) localization can be implemented with regular smartphones without the need for additional hardware for the client. In the future, opportunities for precise localization for both indoors and outdoors can be provided by RSS ngerprinting in very dense 5G networks [15]. In RSS ngerprinting a SP records RSS values from multiple locations in the area covered by the localization service and stores them in a database in a server. When a client wants to perform localization, his/her device measures RSS values and sends them to the server which computes the location based on comparisons with the entries in the database and returns it to the client.

Location data is very privacy sensitive and even little information about peoples' locations allows to identify them [16]. In the shopping mall use case, location information reveals the stores, restaurants, etc., that a customer visits and, consequently, gives out sensitive details about the customer and allows very accurate pro ling. Notice that even a single location query may reveal information that a customer is unwilling to share (e.g., a visit to a speci c shop). WiFi ngerprinting is privacy-violating by nature because the actual localization is performed in the SP's server, which leaks the clients' locations to the SP. Customers would bene t from deployment of a Privacy-Preserving Indoor Localization (PPIL) that prevents the SP from obtaining the customers' locations because this would remove their privacy concerns. Also the SP has incentives to provide PPIL because it would signi cantly increase privacy-aware customers' interest towards an indoor LBS and would help the SP to comply with privacy regulations (e.g., EU GDPR). Hence, a PPIL has potential to bene t both parties, given that it does not imply excessive computation or communication overheads into the system; these are particularly important factors for the clients who are using mobile devices. The SP may additionally require that the scheme can be extended with additional features such

R. Nieminen, while preparing the research work described in this paper, was with Department of Computer Science, University of Helsinki, Helsinki, Finland, e-mail: raine.i.nieminen@iki.
K. Järvinen is with Department of Computer Science, University of Helsinki, Helsinki, Finland, e-mail: kimmo.u.jarvinen@helsinki.

Manuscript date: January 25, 2021.

as (privacy-preserving) location-based targeted advertising and statistics about clients' movements.

Clients' location privacy could be fully protected by sending the database to the clients' devices and performing localization locally. This would also be essentially free in terms of computation and communication overheads. Unfortunately, performing localization in clients' devices is seldom an option in practice because it contradicts the SP's interests. The database is the SP's primary asset as database collection is laborious and time-consuming. A secret database permits the SP to charge from the use of the localization service (either from the owner of the premises or the clients) and, therefore, the SP has economical incentives to hide the database which has often lead to sacri cing clients' privacy. The database may also reveal sensitive details about the infrastructure, e.g., thick doors or walls, which should be kept in secret, especially, from criminals. Hence, PPIL should hide (a) the clients' locations from the SP and (b) the database from the clients. Finally, we emphasize that PPIL solves only a part of all location privacy problems because there can be other ways to track clients' movements, but they are out of the scope of this work.

A few attempts to develop a PPIL scheme for RSS ngerprinting ful lling the above requirements are available in the literature. Li et al. [17] presented a PPIL scheme based on Paillier encryption [18], but severe weaknesses leading to full database recovery were recently found in their system in [19]. Konstantidis [20] used k-anonymity to hide a client's real location trace among k 1 fake traces, but the protection is not very strong because use of auxiliary information (e.g., a building map) may reveal the real trace. Zhang et al. [21] showed a PPIL based on Support Vector Machine (SVM) and Paillier encryption, but [19] showed that it suffers from similar weaknesses as [17]. Yang and Järvinen provide four proposals for PPIL in [19]. The most promising proposal appears to be a hybrid secure two-party computation protocol based on Paillier encryption and Garbled Circuits (GCs), but they provides only a high-level sketch of the scheme without detailed implementation considerations or results. In [22], Järvinen et al. propose a scheme based on secure multi-party computation where the client and server outsource most of the computation and communication to two semi-trusted servers. While their scheme achieves very good performance, the requirement for semi-trusted parties can be a problem in some cases because such trusted parties may not be available. To summarize, the literature still lacks a secure yet ef cient two-party PPIL for RSS ngerprinting.

We provide the following contributions:

- We describe a PPIL scheme based on Yang and Järvinen's high-level proposal from [19]. Our scheme is based on standard cryptographic primitives: Paillier encryption, GC, and One-Time Pad (OTP).
- We present several optimizations to the scheme that considerably improve its performance by reducing both computation and communication overheads.
- We develop a Proof-of-Concept (PoC) implementation of the scheme for Android smartphones and commodity servers that allows us to evaluate the practical feasibility of the scheme.
- Our scheme is the rst two-party PPIL scheme based on RSS ngerprinting that is both secure and feasible for deployment.

The rest of the paper is structured as follows. Sec. 2 surveys the relevant background on indoor localization, cryptographic primitives, and the threat model. We describe our PPIL scheme in Sec. 3. In Sec. 4, we present multiple optimization techniques that are required to make our scheme practical. In Sec. 5, we provide theoretical evaluation of computation and communication overheads. We describe the details of a PoC implementation for Android smartphones and Linux servers in Sec. 6 and provide results from it in Sec. 7. Finally, we draw conclusions in Sec. 8.

2 Preliminaries

This section covers the required background. We start by explaining RSS ngerprinting based indoor localization in Sec. 2.1. Cryptographic techniques are described in Sec. 2.2, 2.3, and 2.4. Finally, we describe our threat model in Sec. 2.5.

2.1 Indoor Localization

Indoor areas, such as airports and shopping malls, require a non-GNSS based localization technique. A common solution for indoor localization is a ngerprint-based localization scheme where an SP holds a database of RSS information pre-measured from prede ned Access Points (APs) in certain locations [23], [24]. Our scheme does not rely on a speci c type of APs, so the source of the RSS is not relevant at this point, but they can be, e.g., from WiFi APs.

For simplicity, the localization process can be divided into two phases:

Training phase. The SP constructs the database D. Firstly, the SP determines a set of APs used in the scheme and de nes a (public) list

$$T_1 = \{AP_j\}_{j=1}^{N}$$

where AP_j is a unique identi er of the j-th AP (e.g., the MAC address) and N is the number of the APs. Next, the SP speci es the locations, where RSS values are going to be pre-measured and de nes a (public or private) ordered list

$$T_2 = \{\ _i\}_{i=1}^{M}$$

where $_i$ is the location parameter at the i-th position (e.g., coordinates) and M is the number of the locations.

Then the SP (or a contractor) visits each location of T_2, measures the RSS $v_{i;j}$ from all the APs de ned in T_1, and constructs a database D from the collected values and lists $T_1; T_2$. An overview of the structure of D can be seen in Tab. 1. D is stored on a server S deployed by the SP.

Location retrieval phase. In this phase, a client C requests its current location from the server S. Firstly, C asks the list T_1 from S, if C does not already have it. Then, C measures the RSS f_j of all the APs in T_1 at its current location and sends these values to S. The server S computes the distances of the received RSS values and the pre-measured RSS values in D

TABLE 1
The structure of the database D constructed during Training phase of the ngerprint-based localization scheme.

i	Location	AP_1	AP_2	...	AP_N
1	1	$v_{1;1}$	$v_{1;2}$		$v_{1;N}$
2	2	$v_{2;1}$	$v_{2;2}$		$v_{2;N}$
⋮	⋮	⋮	⋮	⋱	⋮
M	M	$v_{M;1}$	$v_{M;2}$		$v_{M;N}$

for each location in T_2 and obtains a list of distances $\{d_i\}_{i=1}^{M}$. We use the squared Euclidean distance $d_i = \sum_{j=1}^{N} (f_j - v_{i;j})^2$ and utilize the fact that the formula for d_i can be expanded into three parts [25]:

$$d_i = \sum_{j=1}^{N} f_j^2 + \sum_{j=1}^{N} (-2 f_j v_{i;j}) + \sum_{j=1}^{N} v_{i;j}^2 : \quad (1)$$

We later refer to the three terms in (1) as $_1$, $_2$, and $_3$, respectively. Finally, S nds the indices of the k smallest distances in the list. The method is known as the k-Nearest Neighbor (kNN) [26]. The location of C is computed as the centroid of the locations in T_2 corresponding to the indices.

2.2 Paillier Encryption Scheme

The Paillier encryption scheme [18] is an additive homomorphic probabilistic public-key cryptosystem proposed by P. Paillier in 1999. In the following, we describe the cryptosystem and review certain properties of the scheme. Further details and optimizations are found in [18] and [27].

Key Generation. Let p and q be two randomly selected large prime numbers and $n = pq$. Let B be the set of elements of order n in Z_{n^2}, where is a positive integer, and g a base from B. The public key pk is the pair $(n; g)$ and the secret key sk is $= \mathrm{lcm}(p-1; q-1)$, i.e., the least common multiple of $p-1$ and $q-1$.
Encryption. For a plaintext $m \in Z_n$, the encryption is done as follows. We select a random $r \in Z_n$ and compute the ciphertext c by using the formula

$$c = g^m r^n \mod n^2 \quad (2)$$

with the public key $(n; g)$. Henceforth, Paillier encryption of m is denoted by $E(m)$.
Decryption. Decryption of a ciphertext $c\ (< n^2)$ is performed as follows:

$$m = \frac{L(c \mod n^2)}{L(g \mod n^2)} \mod n ; \quad (3)$$

where is the secret key and $L(u) = \frac{u-1}{n}$. Henceforth, Paillier decryption of c is denoted by $D(c)$.

Paillier encryption is an additively homomorphic encryption scheme and we emphasize the following properties that are used in our scheme:

$$\forall m_1; m_2 \in Z_n \quad \text{and} \quad k \in N$$
$$D(E(m_1)E(m_2) \bmod n^2) = m_1 + m_2 \bmod n \quad (4)$$
$$D(E(m_1)^k \bmod n^2) = k m_1 \bmod n : \quad (5)$$

The reason why the Paillier encryption scheme is used in our localization scheme follows from (4) and (5). Other secure and ef cient additively homomorphic encryption schemes offering the same features could be used as well (see, e.g., DGK [28], exponential ElGamal [29] and the blinded encryption schemes [30]); the choice of Paillier encryption is supported by the facts that it is widely-studied and has an ISO standard (ISO/IEC 18033-6:2019).

2.3 One-Time Pad

OTP is an encryption technique that provides perfect secrecy [31] and it works as follows.

Key Generation. Choose a unique truly random key $R \in Z_n$, where n is a positive integer.
Encryption. For a plaintext $m \in Z_n$, use the key $R \in Z_n$ to compute the ciphertext $c = m + R \bmod n$.
Decryption. Decryption follows similar procedure, namely, $m = c - R \bmod n$.

It is crucial for the security that R is a truly random value of the size of the plaintext and must never be reused.

2.4 Garbled Circuits

GC is a technique for secure computation that was implicitly introduced by A. Yao in [32]. GC can be used for secure two-party computation, i.e., two parties can evaluate a function $f(x; y)$ without revealing the respective inputs $x; y$ to each other or a third party. The idea of GC is to describe the function as a Boolean circuit for the two inputs. The wires of the circuit are assigned with two random s-bit values corresponding to the bit values zero and one, respectively. The gates of the circuit are "garbled" so that their output wire values are encrypted by using the input values as keys according to the truth tables, after which the rows of the tables are randomly permuted. The derived tables form the "garbled" function $\tilde{f}(\tilde{x}; \tilde{y})$, which can be evaluated securely with the garbled inputs $\tilde{x}; \tilde{y}$ to obtain the result.

Yao's original GC proposal is widely considered unpractical due to its high computation and particularly communication overheads, but signi cant improvements have been made during the last decades (see, e.g., [33], [34], [35], [36], [37], [38]). Next, we highlight three state-of-the-art optimizations that have a major impact on the ef ciency of GC based secure computation. In [36], Kolesnikov and Schneider introduced a technique that reduces both computation and communication overhead of each XOR gate of the circuit to zero. In [38], Zahur et al. presented a technique that reduces the communication overhead of gates by 50% (4s to 2s) compared to Yao's original proposal. In [34], Bellare et al. showed that GCs can be constructed by using a block cipher (e.g., AES) with a xed key as the basic cryptographic primitive consequently leading to major speedups in computation overhead (e.g,. by using the AES-NI instruction set extension). Our scheme can be implemented by using any GC framework but, as will be discussed with more details in Sec. 6.2, we use ABY [39] for the PoC. It is a state-of-the-art framework that implements (among others) the optimizations of [34], [36], and [38].

2.5 Threat Model

This section covers privacy threat model for a PPIL scheme. In short, the information of both the server S and the client C needs to be secured and kept private. In the perspective of S this means that an adversary A_C who uses the service as a client should not be able to construct the database D (or a close equivalent D' that allows running a similar service). I.e., A_C should not learn the pre-measured RSS values $v_{i,j}$ of D from the messages sent in the protocol. On the other hand, in the perspective of C, S may be A_S who wants to obtain the locations of its clients and the protocol should reveal nothing about the location of C within the area covered by the service. I.e., A_S should not obtain C's location (the output of the protocol), the RSS values f_j of C's query, or any intermediate values that depend on them (e.g., distances d_i).

A regular fingerprint-based localization scheme (e.g., as described in Sec. 2.1) can easily be made resistant against A_S, if S simply sends D to C, but obviously this directly implies insecurity against A_C. On the other hand, even the simple scheme in Sec. 2.1 is resistant against against A_C, if only the final location is sent to C, but then the protocol fails to protect against A_S. Therefore, we demand that a PPIL scheme must be resistant against both types of adversaries A_C and A_S.

Most of the related work has considered the above privacy requirements under the Honest-But-Curious (HBC) threat model (if any at all). The HBC assumes that both parties faithfully follow the protocol but try to compromise the other party's privacy by using information given by the protocol (e.g., message contents). However, Yang and Järvinen [40] recently argued that HBC is not sufficient to protect PPIL in practice. Indeed, a malicious client A_C can easily deviate from the protocol by manufacturing fake queries (e.g., such that all RSS values are zeros or only one is non-zero), send them to S, and deduct information about D from the response [17], [19]. In most PPIL proposals clients' queries are encrypted, so S has no way to notice such malicious behavior. In fact, practical attacks building on such ideas have been recently presented against several PPIL proposals in [19]. A malicious server A_S, on the other hand, must follow the protocol because any deviation would immediately results in a drop in the quality of localization which would be noticed by C who would probably stop using the service [40]. Motivated by these observations, [40] proposed that a Unilateral-Malicious (UM) threat model should be used for PPIL. In UM, A_C is allowed to deviate from the protocol, but A_S is not. In addition to introducing the UM threat model, [40] also proved that the high-level proposal from [19], which is also the basis of our PPIL scheme, is secure against both A_C and A_S under the UM model. For the sake of clarity, we omit further details and point interest readers to [40] for more detailed descriptions of UM and the security proofs.

3 Privacy-Preserving Localization

This section introduces the PPIL scheme based on a high-level idea given in [19]. Basically, the scheme uses the Paillier encryption scheme to hide the clients' RSS values from the server. The Paillier scheme cannot protect the database by itself. Therefore, each distance is masked with a unique random value, which protects the database and prevents the attack from [17]. The mask is removed and the kNN is run within a GC. Although, the GCs could be utilized for a complete location retrieval without the need of the Paillier scheme, the communication overhead would be enormous [19].

3.1 Description of the Scheme

The following introduces the details of the scheme.

Training phase. This phase is exactly the same procedure explained in Sec. 2.1. The outcome is the same database D as in Sec. 2.1 and it is stored on a server S in plaintext. No interaction with any client is needed.

Distribution phase. The purpose of this phase is to let a client C and the server S share necessary values with each other. Ideally, this phase needs to be done only once with each C. In addition to T_1, T_2 and C's Paillier public key pk (defined in Sec. 2.1 and Sec. 2.2, respectively), fixed values used in GC protocol should be distributed according to the requirements of the specific GC protocol in use.

Paillier phase. The location retrieval starts by C measuring the RSS f_i from every AP in T_1. Then, C computes the following list

$$\{E(2f_j)\}_{j=1}^{N}$$

and $\gamma_3 = E(\sum_{j=1}^{N} f_j^2)$, which are sent to S. Next, S computes the following values for $i = 1, \ldots, M$

$$\gamma_{i,1} = E\left(\sum_{j=1}^{N} v_{i,j}^2\right) \quad (6)$$

$$\gamma_{i,2} = \prod_{j=1}^{N} E(2f_j)^{v_{i,j}} \quad (7)$$

and then computes $D_i = \gamma_{i,1} \gamma_{i,2} \gamma_3$. Due to the homomorphic properties of Paillier encryption (see (4) and (5)), decryption of D_i would yield d_i, where d_i is exactly the distance as in (1). Finally, S masks each D_i with a random value $R_i \in Z_n$. I.e., S computes the following list

$$\{D_i \cdot E(R_i)\}_{i=1}^{M} \quad (8)$$

and sends it to C, who decrypts every item on the list and obtains

$$\{d_i + R_i\}_{i=1}^{M}.$$

GC phase. In this phase, C uses a garbled circuit (see Sec. 2.4) to remove the masks and to compute the kNN in the following way

1) S constructs a Boolean circuit that takes two lists X, Y both containing M values of size l', where l' is the maximum bit-length of a distance. The circuit computes first $X - Y = \{x_1 - y_1, \ldots, x_M - y_M\}$, and then returns the indices of the k smallest values in the list. This Boolean circuit is then used in GC construction and the GC is sent to C along with the servers garbled input $Y = \{y_1, \ldots, y_M\} =$

$\{\tilde{R}_1, \ldots, \tilde{R}_M\}$, where $\tilde{R}_i$ is the garbled value of the random value R_i.

2) Before C can evaluate the GC, it needs to obtain the garbled values $\tilde{X} = \{\tilde{x}_1, \ldots, \tilde{x}_M\} = \{\tilde{d}_1 + R_1, \ldots, \tilde{d}_M + R_M\}$ for its input X by using an Oblivious Transfer (OT) protocol [41], [42].
3) C evaluates the GC and obtains the list of indices of the k smallest distances. Finally, C deduces the locations from T_2 and retrieves its own location by computing the centroid.

Remark 1. In Step 3 of GC phase, T_2 could be a part of the GC, which could then return the location directly, i.e., the centroid can be computed within the circuit. The GC phase may also include Step 4 containing additional service related material, such as promo codes or discount vouchers, allowing location-based targeted marketing without losing privacy and practicality.

3.2 Security

The security of our scheme relies on commonly known cryptographic protocols, namely, Paillier encryption, OTP and GCs (see Sec. 2.2, 2.3 and 2.4, respectively). In this section, we take a closer look on how the system protects against the attacks described in Sec. 2.5 and what assumptions need to be made. The high-level proposal from [19] was shown to be secure under the UM threat model in [40]. Because our scheme follows the high-level proposal, this proof holds also for our scheme and we claim that our scheme is secure under the UM model. In the following, we provide an informal rationale behind this claim and refer interested readers to [40] for a formal treatment.

3.2.1 Privacy of the Client's Location

The client's precise location information (RSS values) is encrypted during the whole Paillier phase and is safe from any adversary assuming that Paillier encryption is secure. After a client decrypts the received masked distances, GC phase begins. Therefore, the client's location information is still vulnerable at this point and we must be able to trust on the security of the GC protocol. Precisely, the OT protocol must ensure that A_S is not able to reveal any information of the client's input for the GC. There are many well studied OT protocols, which are believed to be secure (see [41] and [42]).

3.2.2 Privacy of the Server's Database

Without the random mask R on the distances, the server's database D is leaked to the client during the Paillier phase response after N queries. The attack is fully explained in [17], but basically the client simply solves a set of linear equations to obtain the exact D. Therefore, it is essential that the masking secures the distance values from adversaries. This is the case because the random mask acts as an OTP. The mask is removed during GC phase inside the GC. Hence, the secrecy of D relies also on the security of the GC protocol. GCs are believed to be secure when proper label/key sizes and secure symmetric-key algorithms are used in the construction and evaluation.

4 Optimizations

This section covers several techniques to decrease the computational and communication overheads. Without these techniques, the scheme described above would be unpractical in areas, where parameters (particularly N or/and M) are large. In practice, both N and M depend on the size of the building covered by the service but typically $M > N$. Some public databases are available[1] and, e.g., $M = 505$ and $N = 241$ for the TUT measurement data of a four-story office building. However, it is possible that N could be significantly pruned in such databases without significant reductions in localization accuracy by removing less significant APs and possible duplicates (e.g., one AP with several MAC addresses).

4.1 Client Pre-computation

The Paillier encryption (see (2)) contains two modular exponentiations and one multiplication. Especially, $r^n \bmod n^2$ is computationally involving for every encryption, since r and n are always large. However, that part of the encryption is independent of the message m and can be computed in advance. For each encryption, r needs to be a fresh random value, and therefore multiple values should be pre-computed into a stack.

With the pre-computation of $r^n \bmod n^2$, the encryption complexity depends on $g^m \bmod n^2$. On C's side, the message m is always related to an RSS value, which is normally bounded to a small value. Technically, C needs to encrypt $\sum_{j=1}^{N} f_j^2$ and every $\{2f_j\}_{j=1}^{N}$. However, later we will see that $\sum_{j=1}^{N} f_j^2$ does not need to be encrypted. Therefore, C can store $g^{2f} \bmod n^2$ for each possible RSS value. Consequently, each encryption is only a multiplication of two pre-computed values.

Each query requires $N + 1$ encryptions on C's side. Hence, a small stack of pre-computed $r^n \bmod n^2$ values runs out of values after a few queries. On the other hand, constructing a very large stack might not be feasible on C's side due to the time and power consumption. The overhead of pre-computations could be reduced, e.g., by performing them in the nighttime while the device is charging, but this must comply with the background processing policies of the mobile operating system. In a practical scenario, the stack size should be at least $N + 1$ and the stack could be initialized before the actual query is made. This reduces the online location retrieval time but prevents C from making a new query immediately after the previous one. This is acceptable in many practical scenarios (see, e.g., Ex. 4.3).

4.2 Server Pre-computation

If storage space is not a problem, every $\epsilon_{i,1}$ for $i = 1, \ldots, M$ (see (6)) can be pre-computed for each client after Distribution phase. This reduces the computational overhead tremendously while computing the encrypted distances D_i, since we can avoid M encryptions.

1. UJIIndoorLoc Data Set: https://archive.ics.uci.edu/ml/datasets/ujiindoorloc; TUT Indoor WLAN measurement data: http://www.cs.tut./tlt/pos/MEASUREMENTS_WLAN_FOR_WEB.zip

4.3 Server Computation Improvements

A dynamic programming style technique can be applied for computation of $_{i;2}$ with (7) to reduce the computational overhead. Firstly, we make an observation that each $v_{i;j}$ is normally a small number. E.g., if the maximum bit-length of RSS is 4, then $v_{i;j}$ 2 f 0; 1; : : : ; 15g. Let $v_{max;j}$ denote the maximum value of $v_{i;j}$ for each $j = 1; : : : ; N$ (the maximum RSS in each column of the database), then we compute the lists $L = f1; E(\ 2f_j); E(\ 2f_j)^2; : : : ; E(\ 2f_j)^{v_{max;j}} g_{j=1}^{N}$. Now every possible $E(\ 2f_j)^{v_{i;j}}$ value is computed and can be derived from the lists during the computation of each $_{i;2}$ for $i = 1; : : : ; M$. This technique is shown in Alg. 1 and it prevents the server from computing the same exponentiation multiple times.

Algorithm 1: An optimization technique for $_{i;2}$

```
Data: E(  2f_1); : : : ; E(  2f_N ); D
Result: L
for j = 1 to N do
    v_max    0;
    for i = 1 to M do
        v    D[i; j ];
        v_max    max(v; v_max );
             1;
    list[0]    ;
    for i = 1 to v_max do
                 E(  2f_j );
        list[i]    ;
    L[j ]    list;
```

4.4 Distance Packing

S has to send M ciphertexts to C, which increases the communication overhead tremendously in a real life setting, where M is large. To overcome this issue, S can pack multiple distances into one ciphertext using the packing technique from [25]. Firstly, we need to deduce the bit-length of a distance d_i, which can be derived from (1). Let l be the maximum bit-length of a RSS value, then

$$l^0 = \left\lceil \log_2 (2^l \quad 1)^2 \ N \right\rceil \tag{9}$$

is the maximum bit-length of d_i for every $i = 1; : : : ; M$.

Essentially, by packing we mean that multiple distances are aligned in one larger value by linearly shifting them with a proper constant value. We call the result a package regardless of whether the distances in it are in encrypted form or not. The length of a package is the number of distances it contains.

The maximum bit-length of the plaintext in Paillier encryption is $= \lfloor \log_2(n) \rfloor$. To avoid the possibility of an over ow (with high probability) resulting in a modular reduction modulo n during decryption after R is added into the package (cf. (8)), we leave some of the most signi cant bits to zero on purpose by packing fewer distances into one plaintext that what would t there. Let t be the number of distances we pack in one package and T the total number of packages needed to transmit all M distances. After S has computed D_i for all $i = 1; : : : ; M$, it computes the packages as follows

$$\left(\prod_{i^0=1}^{t} D_{t(k \quad 1)+ i^0}^{2^{(i^0 \quad 1) l^0}} \right)_{k=1}^{T} :$$

Example 4.1. Let $M = 7$ and so we have the encrypted distances $D_1; : : : ; D_7$. We assume that $l^0 = 13; \ = 2048$ and $t = 7$. Thus, $T = 1$ and we can t all the distances in one package by computing $\prod_{i=1}^{t} D_i^{2^{(i \quad 1) l^0}}$. Due to the homomorphic properties, the effect on the plaintext is the following

$$\sum_{i=1}^{t} d_i \ \ 2^{(i \quad 1) l^0} :$$

The distances line up in the plaintext as shown in Fig. 1.

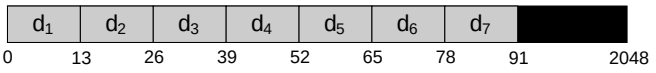


Fig. 1. Example of packing

The masking is done in a similar fashion to (8), but only T masks are needed. The random mask should be chosen uniformly at random from Z_n to avoid any information leaking about the distances after the decryption.

If the distances cannot be divided exactly into T packages of length t, the remaining distances are distributed evenly in the packages, i.e., we have that $M = T \quad t +$ and $T = \lfloor M=t \rfloor$. The best choice for T (or t) can be derived from the minimum point of a cost function that evaluates the complexity of the processes related to the packing, namely the packing itself, the random masking, the overhead of sending the packages to C and the decryption at the C's side.

For every distance d_i, the plaintext of $_3$, namely, $_3 = \sum_{j=1}^{N} f_j^2$, is a positive constant that only increases the value of every d_i. Therefore, the kNN gives the same result, even if $_3$ is left out. However, the distances d_i are not necessary positive anymore, which is problematic, since we are using modular arithmetic. To overcome this problem, the client adds $_3$ to the decrypted masked distances after the decryption to ensure the positivity of each distance. With the packing technique, $_3$ needs to be added to every distance in a package.

Example 4.2. Let us use a similar setup as in Ex. 4.1. Without $_3$ the plaintext would be $\sum_{i=1}^{t} (\ _1 + \ _2) \ 2^{(i \quad 1) l^0} + R$. Therefore, we need to add $\sum_{i=1}^{t} \ _3 \ 2^{(i \quad 1) l^0}$ to obtain

$$\sum_{i=1}^{t} (\ _1 + \ _2 + \ _3) \ 2^{(i \quad 1) l^0} + R = \sum_{i=1}^{t} d_i \ 2^{(i \quad 1) l^0} + R :$$

Remark 2. Since $_3$ is eventually added to the package(s), the packing cannot be denser, i.e., l^0 must be computed as before (see (9)).

4.5 Masking Pre-computations

The encryption of R_i could be done after Distribution phase by pre-computing them into a stack for each client. However, the number of needed encrypted masks is unknown, since it depends on the number of a particular

client's queries which could be hundreds per day. This technique used with the packing reduces the masking process down to only T multiplications. However, it is difficult to justify the practical feasibility of this technique.

Example 4.3. Let us imagine a shopping mall that offers a PPIL service by using our scheme. We can store a certain number of encrypted masks for each client and refill this storage, e.g., during the nighttime (at least for several thousands of clients). If such a storage is not used or it runs out of pre-computed values, the service may limit the frequency at which the client can make queries (e.g., only every 5 seconds) and use the intermediate time for filling the stack of pre-computed values. This would not degrade the quality of service significantly. This also allows the client to perform the client-side pre-computations discussed in Sec. 4.1. The only difference is that the server must serve several clients simultaneously and perform multiple pre-computations in the same time where a client performs only its own pre-computations.

4.6 Garbled Circuit Pre-computations

Since GC phase is a distinct part in the scheme, it is difficult to do any specific optimizations. However, since the structure of the circuit is known, the server could construct it in advance. The problem is that the GC needs to be unique for each query and the size of one GC is already large (even several megabytes).

In practice, the new GC for the next query can be constructed right after the current query. This technique follows the same idea as the Paillier phase optimization techniques with pre-computation, where we assumed that there is a long enough delay between multiple queries. Furthermore, the new GC could be sent to the client right after the construction before the actual query is made. This may give a notable saving in time from making the location query to receiving the location, especially, in slower networks.

5 Theoretical Evaluation

We denote the computational cost of a modular multiplication with Mult-C(m) and an exponentiation with Exp-C$(a; m)$, where a is the exponent and m is the modulus. The square-and-multiply algorithm gives

$$\text{Exp-C}(a; m) = 1.5 \log_2 a \quad \text{Mult-C}(m) \tag{10}$$

on average (the exact cost depends on the Hamming weight of a).

With Crypt-C(s) we denote the computational cost of a decryption in the garbled circuit evaluation, where $s = \lceil \log_2 w \rceil$ is the label size (in bits). The computational cost depends on s and the garbled circuit encryption function.

5.1 Computational Overhead

The computational overhead of C during Paillier phase is

$$\begin{aligned}&\text{C-Mult-C}^{\,2}(N; M)\\ &= N \quad \text{Mult-C}(n^2) + T \quad 3{=}2 \log_2 n \quad \text{Mult-C}(n)\end{aligned}$$

with the pre-computation technique, which decreases the encryption step down to N multiplications modulo n^2. The rest of the overhead comes from the decryption step, which involves T decryptions. The cost of one decryption is approximately $3{=}2 \log_2 n$ multiplications modulo n with the state of art optimizations [18], [27].

The following equation

$$\text{Mult-C}(n^2) = 4 \quad \text{Mult-C}(n) \tag{11}$$

gives the complexity relation of a schoolbook multiplication[2]. It is used to simplify the overheads and we get

$$\text{C-Mult-C}^{\,2}(N; M) = (N + T \quad 3{=}8 \log_2 n) \quad \text{Mult-C}(n^2) :$$

The computational overhead of S for Paillier phase is

$$\begin{aligned}&\text{S-Mult-C}^{\,2}(N; M)\\ &= (N \quad (2^l \quad 2) + M \quad N_a + M \quad ((t \quad 1) l'{=}2 + 1) + T)\\ &\qquad \text{Mult-C}(n^2)\end{aligned}$$

where N_a denotes the average number of non-zero RSS values in each row of D, which basically stands for the average number of available APs at each location[3]. The first part of the complexity comes from Alg. 1, when we assume the worst case, i.e., each column contains every possible RSS value. Finally we can compute every D_i with the list L using $M \quad N_a$ multiplications modulo n^2. The rest of the overhead comes from the packing (see Sec. 4.4) and from the masking, which is only T multiplications with the pre-computations (see Sec. 4.5).

Finally, we omit the multiplication complexity and obtain

$$\text{Client-C}(N; M) = N + T \quad 3{=}8 \log_2 n \tag{12}$$

$$\begin{aligned}&\text{Server-C}(N; M)\\ &= N \quad (2^l \quad 2) + M \quad (N_a + (t \quad 1) l'{=}2 + 1) + T :\end{aligned} \tag{13}$$

Now we can plot the overheads to find out which party is computationally more involving in our scheme. Firstly, we fix several parameters. Let $l = 4$ and by using (9), we get l' as a function of N. For simplicity, we set $N_a = 0{:}15N$; $n = 2^{2048}$ and $t = 25$, which makes $T = \lceil \frac{M}{t} \rceil$ dependent only on M. Now both (12) and (13) depend only on N and M. The overheads with these assumptions are shown in Fig. 2. Clearly, S has a higher computational overhead than C. However, the overhead can be shifted by altering t. Here we suggested that $t = 25$ is a good overall upper bound for the number of distances in packages, but other values may be more appropriate in certain applications.

The computational overhead of GC phase comes only from the GC execution on C's side, when the pre-computation techniques are used (see Sec. 4.6). The execution complexity depends on the number of AND gates in the circuit. It can be estimated using information available in [33] that the subtraction and the kNN require approximately $M \quad l' =$ AND gates and $3k$ AND gates, respectively. Therefore, we get the following overhead

$$\text{GC-C}(N; M) = (3k + 1) \quad \text{Crypt-C}(s) : \tag{14}$$

2. There exists faster variants of a long multiplication, such as the Toom-Cook algorithm [43].

3. In practice, about 85% of the values in the database are normally zeros (see, e.g., [44]), which means that $N_a \quad 0{:}15N$.

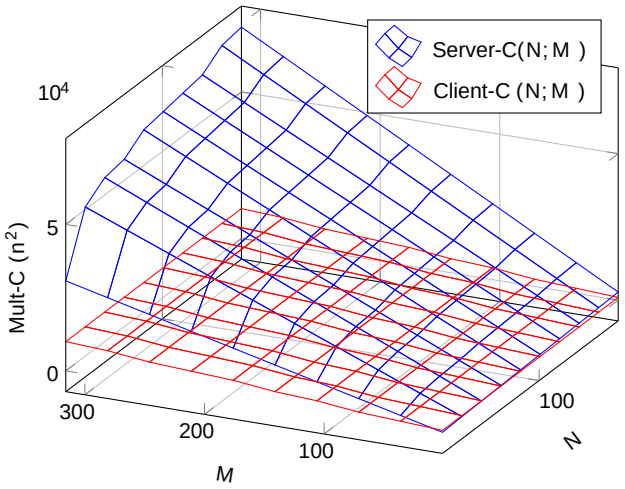


Fig. 2. The number of multiplications modulo n^2 needed for Paillier phase separately for S and C, when $l = 4$, $N_a = 0:15N$, $n = 2^{2048}$ and $t = 25$.

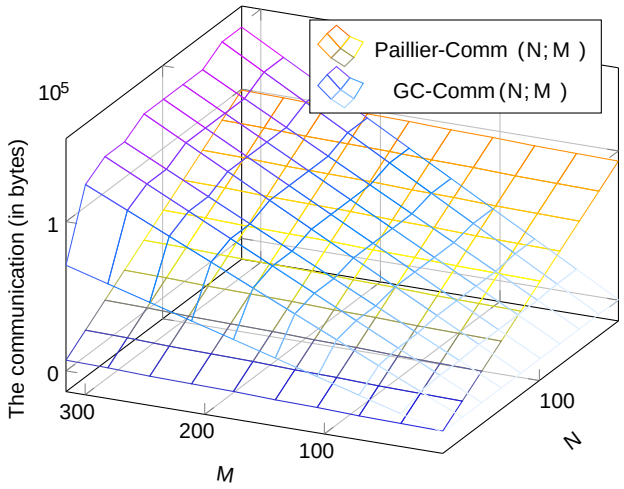


Fig. 3. The communication overheads in bytes for Paillier phase and GC phase separately, when $l = 4$; $n = 2^{2048}$; $t = 25$ and $s = 112$.

In general, Mult-C (n) is more complex than Crypt-C (s) (e.g., AES), but the exact relation depends on the computing platform. The cost of Crypt-C (s) can be often reduced significantly even with commodity hardware by utilizing speci c instructions such as Intel's AES-NI. Accelerating Mult-C (n) is more complicated, but speci c hardware accelerators (e.g., with FPGAs) can be used on the server side, and they may allow signi cant speedups for these operations.

5.2 Communication Overhead

The communication overhead (in bits) of Paillier phase is

$$\text{Paillier-Comm}(N; M) = (N + T) \log_2 n^2 \quad (15)$$

and during GC phase the overhead is

$$\text{GC-Comm}(N; M) = 2s \quad M \quad l^0: \quad (16)$$

We assume that the GC and S's (garbled) input have been sent in advance and that the OT pre-computations are done. Fig. 3 depicts the overheads when N and M are increased with xed parameters $l = 4; n = 2^{2048}; t = 25$ and $s = 112$ Normally, the communication overhead of GC phase becomes dominant.

The communication overhead re ects to the location retrieval time, since the data transfer takes some time. Especially with low bandwidth, the delay might be signi cant. For some clients, the communication overhead might be an issue also due to data transfer costs. Therefore, we also want to see the total communication overhead for each query. The overheads of different objects are gathered into Tab. 2. It is easy to see that the size of the garbled circuit becomes dominant quickly, especially with large k, when M increases. The growth is also linear, which makes extrapolation easy.

6 Implementation

In the following, we give an overview of the PoC implementation of our PPIL scheme introduced in Sec. 3. The client side is programmed to an Android device [4] with the Android Studio[5]. The server side is not restricted to any particular operating system, but we use a GNU/Linux Ubuntu server [6]. The details of the hardware used for evaluating our PoC implementation were Exynos 7420 with 8 CPU cores and 2.74 GB of RAM running at 1.5 GHz for the client side and Xeon E5-2697 v2 with 4 CPU cores and 8.17 GB of RAM running at 2.7 GHz for the server side.

6.1 Paillier Phase

In this section, we concentrate on the implementation aspects of Paillier phase programmed in Java[7]. The implementation follows the description given in Sec. 3 with the optimization techniques from Sec. 4.

The Paillier encryption scheme implementation follows Sec. 2.2 with the state-of-the-art optimization techniques of [18] and [27]. The random numbers are generated with Java class SecureRandom[8], which provides a cryptographically strong random number generator that complies the speci - cation FIPS 140-2.

Modular arithmetic relies on the Java class BigInteger[9]. Random prime numbers of speci c length are generated with the SecureRandom. The most important method is the modPow(..) from the BigInteger class, which computes an exponentiation with a modulus. The method uses the sliding window techniques [45] and the Montgomery domain [46], which improves the ef ciency of an exponentiation slightly more than we estimated in (10).

Remark 3. The modPow(..) is not a constant time algorithm by default, but we ignore this fact since our implementation is a PoC. However, a constant-time exponentiation should be used in practice to prevent timing attacks [47]. In our scheme, the exponents are

4. Samsung S6 SM-G920F running Android 7.0

5. https://developer.android.com/studio/
6. https://www.ubuntu.com/server
7. https://www.java.com/en/
8. https://docs.oracle.com/javase/8/docs/api/java/security/SecureRandom.html
9. https://docs.oracle.com/javase/8/docs/api/java/math/BigInteger.html

TABLE 2
The total communication overhead of the scheme.

Paillier phase Ciphertexts	GC phase OT	Pre-computation Garbled circuit	(Online) S's input	Of ine OT
$(N + T) \log_2 n^2$	2s	$(6k + 2)$ s	s	$4s + 6s^2$

often small (e.g., a RSS value). Hence, a global constant-time exponentiation routine allowing exponents up to the size of n would slow down these computations tremendously. However, because the maximum values of the exponents are known in every exponentiation, tailoring constant-time exponentiation routines for speci c exponent lengths solves this problem.

A multiplication of two BigIntegers follows the Toom-Cook algorithm [43], which is faster than a regular multiplication for large numbers. We store pre-computed values in a ConcurrentLinkedQueue [10], which is thread-safe meaning that we can access the queue safely with multiple threads. Indeed, we use the queue as a stack for the pre-computations.

The communication between a client C and the server S is implemented with Java network sockets. Java objects are transferred between C and S containing the necessary values. The (socket) connection stays open during the computations at S and is closed after C has received the encrypted and masked distances from S. The same connection cannot be used in GC phase, which is a slight drawback in our PoC implementation (see the reason in Sec. 6.2). Ideally, one connection should be kept open for the whole duration of a location retrieval to reduce the number of connection calls that increase unnecessary routing and possible handshakes.

6.2 GC Phase

The implementation of GC phase is separated from Paillier phase, since we take advantage of the framework for ef - cient mixed-protocol secure two-party computation called ABY [39] available online at GitHub [11]. Therefore, the socket connection of Paillier phase mentioned previously cannot be used here. Overall, the implementation of GC phase relies fully on ABY and its functionality, which is used in a black box manner. ABY was chosen because it is a state-of-the-art framework for secure two-party computation.

ABY provides three secure computation schemes based on Arithmetic sharing, Boolean sharing and Yao's garbled circuits. We use Yao's garbled circuits. The OT extension implementation of [42] is used within ABY. ABY requires that the pre-computations must be done during the current query after Paillier phase , which means that they do not give any advantage. This is a practical drawback of our PoC, but this can be xed in production level code. Nevertheless, ABY still allows us to analyze the exact overheads of our scheme even from this PoC implementation (see Sec. 7.3).

The rst subtraction ("the mask removal") is dif cult to implement with the packing technique, since we need to do a subtraction of two large numbers which is not directly supported by ABY and would result in an increase in the size of the GC. To overcome this, we slice this subtraction into individual 16-bit subtractions and tolerate the small errors caused by the lack of carry propagation between the 16-bit subtractions. For simplicity, we x the bit-length of a distance to 16 bits in our implementation, i.e., $l' = 16$ regardless of l and N. Following the recommendations from [48], we x $l = 4$ resulting in an upper bound of 291 for N (due to (9)). This means that for smaller values of N, there will be "free space" (zero bits) after each distance in the package, which in turn means that the chance of the carry-bit error decreases.

TABLE 3
Security levels and corresponding parameter bit-lengths.

Security Level (s)	Paillier (n)	Garbled circuits (w and c)
112	2048	112
128	3072	128

7 Practical Evaluation

In this section, we present the computational and communication overheads of our scheme obtained with measurements from the PoC implementation discussed in Sec. 6. We start by analyzing the practicality of the scheme in real life environment in Sec. 7.1. Next, we construct arti cial databases of different sizes and use them in Sec. 7.2. This allows us to test the implementation for different values of N and M. Finally, we give the overheads of the distinct steps of our scheme in Sec. 7.3.

We provide experimental results for our PoC implementation for cryptographic security levels of 112 and 128 bits. We have gathered parameter sizes of the cryptographic primitives with the corresponding security levels in Tab. 3 according to the guidelines of [49] and [50].

7.1 Real Life Experiment

We experimented our PoC implementation in one building at the university campus. There were 17 Wi-Fi and 17 LTE APs located within the test area. We used three different databases (constructed by us) which each had $M = 76$ reference locations. The databases were constructed by measuring RSS from only the Wi-Fi ($N = 17$), only the LTE ($N = 17$) and both the Wi-Fi and LTE APs ($N = 34$).

We measured the total duration of the location retrieval, which is the time of Paillier and GC phase together, while walking in the building. The parameter values for our implementation during this experiment are gathered in Tab. 4. With this setting, we observed that the location retrieval time is 2:208 seconds on average. We will investigate the time spent in each step precisely in Sec. 7.3, but we mention already here that the OT pre-computations (which are mostly independent of N and M) took 1:032 seconds in

10. https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentLinkedQueue.html
11. https://github.com/encryptogroup/ABY

TABLE 4
The parameter values for our real life experiment.

Security Level (s)	N	M	k	l	l'	T	t
112	17 or 34	76	1	4	16	1	76

TABLE 5
The number of packages T when M is given.

(a) s = 112

M	100	200	300	400	500	600	700	800
T	10	20	25	33	41	50	50	50

(b) s = 128

M	100	200	300	400	500	600	700	800
T	5	15	20	25	33	40	50	50

our experiment (see Tab. 7(a)). In this experiment, the pre-computations had to be included in the online phase since our PoC implementation uses ABY as a separated part as explained in Sec. 6.2. Therefore, the location retrieval time could be reduced down to around one second with a more compatible GC implementation.

Communication overhead is constant for each query. The measurements were done with vnstat [12] at the server. To avoid measurement errors, the following values are averaged over 10 queries. Each query required about 90 KBs of data to be send to the server (uplink) and 327 KBs to be received (downlink), when $N = 17$. When $N = 34$, the communication overhead was 103 and 334 KBs for the uplink and downlink, respectively. Theoretically, the only difference should have been in the uplink data, where additional 17 ciphertexts (8.7 KBs) are sent but small measuring errors may have occurred, particularly, because M is small. The theoretical communication overheads (see Tab. 2) are slightly optimistic (around 282 KBs in total) compared to the practical ones, but the order of magnitude is correct.

7.2 Arti cial Databases

In this section, we form databases of different sizes containing randomized RSS entries of bit-length $l = 4$. We make the databases more realistic by setting most of the entries to zero; more precisely, we set $N_a = 0.2N$.

7.2.1 Paillier Phase

We generate several databases with xed N and M and take a closer look at the computational and communication overheads during Paillier phase. We choose T according to Tab. 5, where we have taken balancing aspects (see Sec. 4.4) under consideration and set T 50 to limit the communication overhead.

The location retrieval times are shown in Fig. 4 for both security parameters 112 and 128. The pre-computation techniques from Sec. 4 are applied, but we have limited the number of client's encryption pre-computations to 50. Consequently, the encryptions are almost free when N 50. On the other hand, the client must perform 50 or 100 online encryptions for $N = 100$ and $N = 150$, respectively, and the total time increases dramatically. We could allow the client to do N encryption pre-computations, but the above procedure allows us to evaluate the overheads of online encryptions and pre-computations.

The security parameter s has a large impact. Fig. 4(a) and 4(b) show that the increased complexity of encryptions and decryptions leads to signi cant increases in the location retrieval times for $s = 128$ compared to $s = 112$ with larger N. When $s = 128$, our scheme becomes impractical for large N and M. However, with the pre-computations ($N = 50$) and $s = 112$, the time stays in decent limits even for large areas with many reference locations (large M).

A detailed discussion of the communication overhead of Paillier phase is omitted here. Nevertheless, we mention that the total communication per query is at most some hundreds of KBs even in large settings.

7.2.2 GC Phase

The computational overhead of GC phase is discussed more closely in Sec. 7.3. Here we proceed to analyze Fig. 5 showing the communication overhead of GC phase. The value of N is irrelevant for GC phase because we used a xed l'. We observe that the uplink communication remains almost constant for both security levels. According to Fig. 5(a) and 5(b), the client sends on average 71.44 and 116.31 KBs with $s = 112$ and $s = 128$, respectively.

The communication overhead is dominated by the downlink transfer(s), notably, when M and/or k are large. The security level does not have a signi cant effect, as shown in Fig. 5(c) and 5(d). The overhead increases linearly with M and k.

7.3 Precise Overheads of Phases

This section breaks our scheme into smaller steps and examines the overhead of each step and its effect to the total overhead. In addition, we see how much “unnecessary” online overhead the ABY implementation creates.

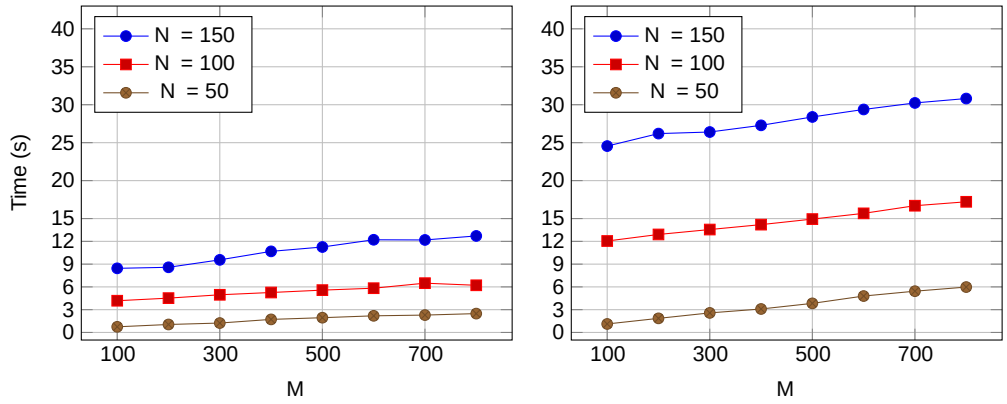
7.3.1 Paillier Encryption with Java

We chose the parameters $N = 50$, $M = 150$, $s = 128$ and $T = 10$ for our test case and collected the result in Tab. 6. The computational overhead is separated into six steps: namely, Encryption, Decryption, Distance, Packing, Masking and Pre-encryption, shown in Tab. 6(a).

The steps involving encryption, speci cally, Encryption and Masking, become negligible with pre-computations computed already after the previous query. The pre-computations are useful because they reduce the online location retrieval time, i.e., the delay that a client must wait after “pushing the button” until nally obtaining the location. The client's (parallelized) pre-computation took 2.447 seconds, which is the time that the client needs to wait before making a new query. This delay between queries is acceptable in many practical settings (see Ex. 4.3), but it could become infeasible in buildings with hundreds of APs.

The Distance step stands for the squared Euclidean distance calculation with the ciphertexts at the server side. It also involves encryptions but they can be avoided with pre-computations (see Sec. 4). Even with Alg. 1, the cost of this step is 28.3% of the total overhead.

12. https://humdi.net/vnstat/



(a) s = 112 (b) s = 128

Fig. 4. The location retrieval time of Paillier phase , when s = 112 ; 128 and N = 50 ; 100; 150. The value for T follows from Tab. 5. The number of client's encryption pre-computations is limited to 50.

TABLE 6
The overhead of Paillier phase with N = 50 , M = 150 , s = 128 and T = 3 .

(a) Computational

Step	Time (ms)
Client	
Encryption	7.77
Decryption	299.07
Total	306.84
Server	
Distance	334.55
Packing	535.91
Masking	3.82
Total	874.28
Pre-computation (Client)	
Pre-encryption	2446.74

(b) Communication

Step	Uplink (bytes)	Downlink (bytes)
For each new query		
Paillier phase	41984	6144

TABLE 7
The overhead of GC phase , when N = 50 , M = 150 , s = 128 and k = 3 .

(a) Computational

Step	Time (ms)
One-time expense	
Init	0.73
CircuitGen	0.07
Network	313.07
BaseOTs	1032.17
Total	1346.04
For each new query	
OTExtension	23.04
Garbling	48.38
Online	294.89
Total	366.31

(b) Communication

Step	Uplink (bytes)	Downlink (bytes)
One-time expense		
BaseOTs	49958	49956
For each new query		
Setup	43256	916881
Online	411	147500
Total	43667	1064381

The rest of the computational overhead comes from the Decryption and Packing steps. The overhead can be shifted between these steps and, consequently, between the client and server, as explained in Sec. 4. In our caseT = 3 and the packing consumed 0.237 seconds more time than the decryptions. Nonetheless, the cost of these steps is dominant emphasizing the importance of wise packing in our scheme.

The communication overhead of Paillier phase is straightforward as shown in Tab. 6(b). The size of one ciphertext is 768 bytes, when s = 128. We compare this to the measurements and obtain that 41984 N 768 = 3584 bytes (uplink) and 6144 T 768 = 3840 bytes (downlink), which indicates that the constant communication overhead produced by the Java object is about 4 KBs.

7.3.2 Garbled Circuits with ABY

We investigate the precise overheads of GC phase by utilizing the benchmarking routines of ABY on the server side. The benchmarking is shown in Tab. 7, where we chose the parameters to be N = 50, M = 150, s = 128, and k = 3 .

The results for the computational overhead are gathered in Tab. 7(a), where we have separated the one-time expenses, i.e., the expense, when two parties connect for the rst time, and the expenses required for each query. We observe that 78:6% of the time goes to the one-time expenses and, as a consequence, the actual computational overhead is only 0:366 seconds per query.

The overhead of the initialization (“Init”) and circuit generation (“CircuitGen”) steps are negligible, but the

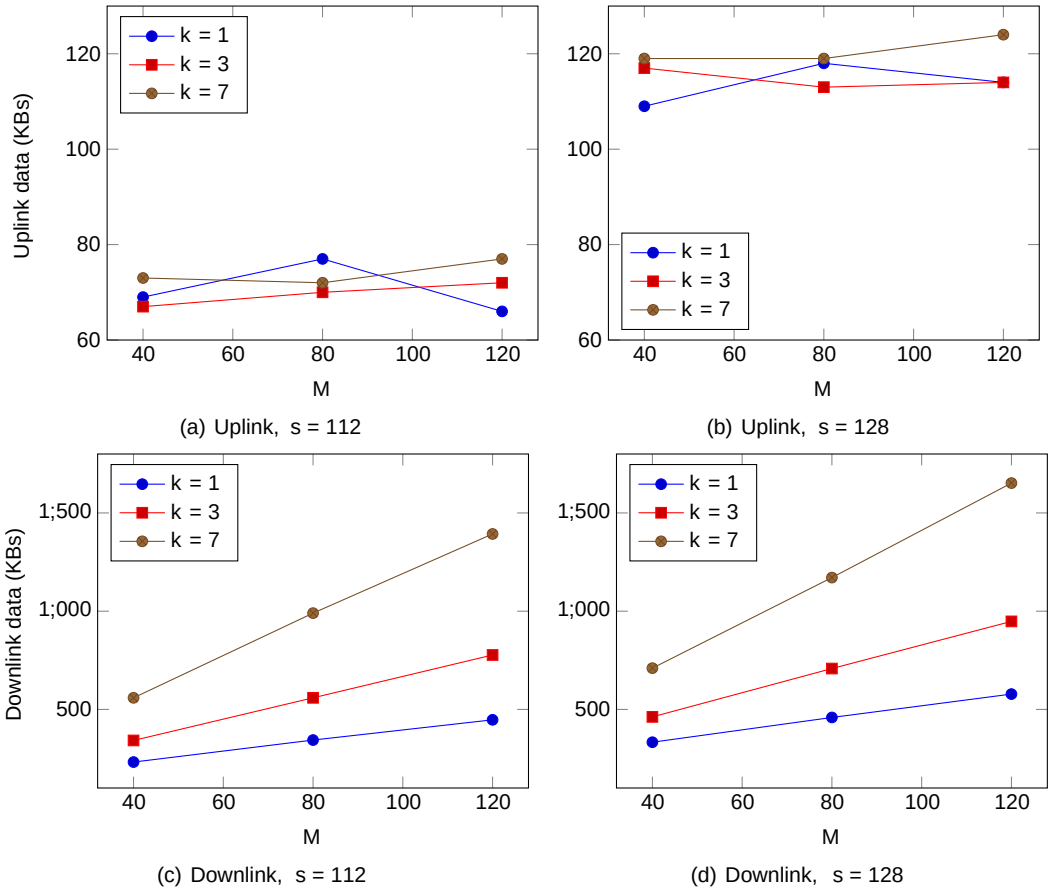


Fig. 5. The amount of data transferred during GC phase , when s = 112 ; 128 and k = 1 ; 3; 7.

"BaseOTs" step takes over a second to complete. This is not surprising since it requires expensive public-key operations. Hence, it is justi ed to say that the location retrieval time in our real life experience (see Sec. 7.1) would be only about one second with a better integrated GC implementation.

The network step of Tab. 7(a) includes the time the server waits the client to connect. With our implementation, the network time includes the time of the Paillier phase response and decryption since the server starts the GC phase immediately after the masking is done. The decryption takes 0:097 seconds (when T = 1) and we can estimate that the response takes roughly 30 ms. This gives the total time of GC phase with our implementation, namely, 1346:04 + 366:31 97 30 = 1585:35 ms. We measured the total time also in the Android application and obtained 1584 ms, which is very well aligned with the benchmarking of ABY.

The communication overheads are shown in Tab. 7(b). The only one-time expense comes from the "BaseOTs" step, which requires 49:96 KBs in both directions. Most of the overheads for each query come from the setup step, which consists of the OTExtension and Garbling steps. The setup downlink overhead consists of the actual GC and dominates the overall communication overhead. The ABY benchmarks show that there are 28650 AND gates, which means that the circuit size is 2 128 28650=(8 1000) = 916:8 KBs. We can conclude that the downlink communication of OTExtension step is negligible in the setup step but the uplink communication for each query is dominated by the OTExtension step. The online step consists of the server sending its inputs to the circuit to the client. Therefore, the downlink communication is dominant.

The measurements with vnstat show that the server received 117:42 KBs (uplink) and sent 110627 KBs (downlink). This is consistent with Tab. 7(b) having 93:63 and 111434 KBs, respectively.

8 Conclusion

We introduced a PPIL scheme based on secure two-party computation following the sketch in [19]. We proposed several optimization techniques and gave the theoretical overheads for the scheme. Furthermore, we implemented a PoC implementation for a basic Android device and experimented it in a real environment. We measured the practical overheads of the implementation with databases of various sizes. They are small enough to consider the scheme as practical for certain applications. However, it is clear that the scheme cannot be used in every setting where indoor localization is used nowadays or it at least requires some additional measures to t in certain applications.

The idea of using Paillier encryption and garbled circuits for PPIL was shown to be feasible with certain reservations. The advantage of such a system is that the privacy relies fully on well-known cryptographic protocols providing cryptographic guarantees that privacy of the inputs of both parties is protected. The drawback is the increment

in computational and communication overheads resulting in longer response delays, greater power consumption, and bigger data usage per query. The costs of privacy are easily noticeable and increase linearly with the building related parameters (N and M). However, a simple tradeoff between efficiency and clients' location privacy exists by splitting the area covered by the service into smaller sub-areas.

Our scheme is suitable for a service that provides clients with an application allowing them to make explicit location retrieval requests to locate themselves in buildings such as airports, hotels, and shopping malls. In such scenarios, the client can wait for a few seconds to retrieve the location and is unlikely to make several consecutive queries. Our PoC implementation was designed for such a scenario and it was demonstrated to be practical with real life experiments.

The scheme is infeasible when quick tracking of a fast object, such as a vehicle, is required. However, continuous tracking of walking speed movements could still be done by combining our scheme with tracking based on auxiliary information. E.g., an application may refresh the location every 5–10 seconds with our scheme and use, e.g., the motion sensors of a smartphone to track the movements between the queries. The overheads can be mitigated also with high performance devices, but other factors, such as power consumption and heat, may set up new practical limitations. On the other hand, we can safely assume that performance of devices (i.e., smartphones and servers) continues to improve also in the future together with other aspects such as faster communication and better batteries, consequently, increasing the practical attractiveness of our scheme. Nevertheless, it is unlikely that our scheme will ever be feasible, e.g., for autonomous cars.

Topics for future research include at least the following:

- It is essential that APs or reference locations that have no or only minor effect on localization accuracy are pruned from the database to avoid unnecessary overheads. Optimal ways for such pruning and tradeoffs between database sizes and localization accuracy require further research.
- Hardware acceleration of critical operations on the server side can provide significant reductions for the server side delays. The expensive Paillier operations are the primary candidate for hardware acceleration. The GC phase utilizes mostly secret-key primitives and existing instruction set extensions, e.g., for AES can be used in a straightforward manner.
- Other additively homomorphic encryption schemes (e.g., [28], [29], [30], [51]) may perform better than Paillier encryption in certain settings and the use of other encryption schemes is worth investigating.
- SPs require specific incentives to deploy indoor localization schemes and these are often related to customer tracking and targeted advertising. Privacy-preserving location-based advertising (e.g., discount vouchers) may be added to our PPIL scheme, more precisely in the GC. Privacy-preserving collection of statistics about customers movements may also be built on top of our scheme, e.g., by using existing techniques for privacy-preserving data mining (see, e.g., [52], [53], [54]). Nevertheless, specific schemes should be developed to attract SPs to use PPIL schemes in practice.
- Other techniques for indoor localization exist besides RSS fingerprinting (e.g., angle-of-arrival, time-of-arrival, etc.). PPIL solutions for such techniques should be also studied and developed in the future.

ACKNOWLEDGMENTS

This work was funded by the INSURE project (303578) of Academy of Finland. We thank Matthias Senker and Christian Weinert from Technische Universität Darmstadt for helping to compile the ABY framework for Android devices.

REFERENCES

[1] C. K. Chung, I. Q. Chung, Y. H. Wang, and C. T. Chang, "The integrated applications of WIFI and APP used in the shopping mall environment for menber card E-marketing," in Intl. Conf. Machine Learning and Cybernetics (ICMLC), vol. 2, 2016, pp. 671–675.

[2] T. Guan, L. Fang, W. Dong, Y. Hou, and C. Qiao, "Indoor localization with asymmetric grid-based filters in large areas utilizing smartphones," in IEEE Intl. Conf. Communications (ICC), 2017, pp. 1–6.

[3] S. He, W. Lin, and S.-H. G. Chan, "Indoor localization and automatic fingerprint update with altered AP signals," IEEE Transactions on Mobile Computing, vol. 16, no. 7, pp. 1897–1910, 2017.

[4] C. Langlois, S. Tiku, and S. Pasricha, "Indoor localization with smartphones: Harnessing the sensor suite in your pocket," IEEE Consumer Electronics Magazine, vol. 6, no. 4, pp. 70–80, 2017.

[5] A. Yassin, Y. Nasser, M. Awad, A. Al-Dubai, R. Liu, C. Yuen, R. Raulefs, and E. Aboutanios, "Recent advances in indoor localization: A survey on theoretical approaches and applications," IEEE Communications Surveys & Tutorials, vol. 19, no. 2, pp. 1327–1346, 2017.

[6] A. M. Ladd, K. E. Bekris, A. Rudys, L. E. Kavraki, and D. S. Wallach, "Robotics-based location sensing using wireless ethernet," Wireless Networks, vol. 11, no. 1-2, pp. 189–204, 2005.

[7] P. Tao, A. Rudys, A. M. Ladd, and D. S. Wallach, "Wireless LAN location-sensing for security applications," in ACM Workshop on Wireless Security, 2003, pp. 11–20.

[8] A. M. Ladd, K. E. Bekris, A. P. Rudys, D. S. Wallach, and L. E. Kavraki, "On the feasibility of using wireless ethernet for indoor localization," IEEE Transactions on Robotics and Automation, vol. 20, no. 3, pp. 555–559, 2004.

[9] A. M. Ladd, K. E. Bekris, G. Marceau, A. Rudys, D. S. Wallach, and L. E. Kavraki, "Using wireless ethernet for localization," in IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, vol. 1, 2002, pp. 402–408.

[10] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki, "Practical robust localization over large-scale 802.11 wireless networks," in Intl. Conf. Mobile Computing and Networking (MobiCom), 2004, pp. 70–84.

[11] J. Talvitie and E. S. Lohan, "Modeling received signal strength measurements for cellular network based positioning," in Intl. Conf. Localization and GNSS (ICL-GNSS). IEEE, 2013, pp. 1–6.

[12] K. Chawla, C. McFarland, G. Robins, and C. Shope, "Real-time RFID localization using RSS," in Intl. Conf. Localization and GNSS (ICL-GNSS). IEEE, 2013, pp. 1–6.

[13] L. Chen, H. Kuusniemi, Y. Chen, L. Pei, T. Kröger, and R. Chen, "Information filter with speed detection for indoor Bluetooth positioning," in Intl. Conf. Localization and GNSS (ICL-GNSS). IEEE, 2011, pp. 47–52.

[14] A. S.-I. Noh, W. J. Lee, and J. Y. Ye, "Comparison of the mechanisms of the Zigbee's indoor localization algorithm," in ACIS Intl. Conf. Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD). IEEE, 2008, pp. 13–18.

[15] A. Hakkarainen, J. Werner, M. Costa, K. Leppänen, and M. Valkama, "High-efficiency device localization in 5G ultra-dense networks: Prospects and enabling technologies," in IEEE Vehicular Technology Conf. (VTC Fall). IEEE, 2015, pp. 1–5.

[16] S. M. Bellovin, R. M. Hutchins, T. Jebara, and S. Zimmeck, "When enough is enough: Location tracking, mosaic theory, and machine learning," NYU Journal of Law & Liberty vol. 8, p. 556, 2013.

[17] H. Li, L. Sun, H. Zhu, X. Lu, and X. Cheng, "Achieving privacy preservation in WiFi ngerprint-based localization," in IEEE Conf. Computer Communications (INFOCOM), April 2014, pp. 2337–2345.

[18] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in Advances in Cryptology — EUROCRYPT. Springer, 1999, pp. 223–238.

[19] Z. Yang and K. Järvinen, "The death and rebirth of privacy-preserving WiFi ngerprint localization with Paillier encryption," in IEEE Conf. Computer Communications (INFOCOM). IEEE, 2018, pp. 1223–1231, full version: https://eprint.iacr.org/2018/259.pdf.

[20] A. Konstantinidis, G. Chatzimilioudis, D. Zeinalipour-Yazti, P. Mpeis, N. Pelekis, and Y. Theodoridis, "Privacy-preserving indoor localization on smartphones," in IEEE Intl. Conf. Data Engineering (ICDE), May 2016, pp. 1470–1471.

[21] T. Zhang, S. S. M. Chow, Z. Zhou, and M. Li, "Privacy-preserving Wi-Fi ngerprinting indoor localization," in Advances in Information and Computer Security (IWSEC), 2016.

[22] K. Järvinen, H. Leppäkoski, E. Lohan, P. Richter, T. Schneider, O. Tkachenko, and Z. Yang, "PILOT: Practical privacy-preserving indoor localization using OuTsourcing," in IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2019, pp. 448–463.

[23] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in IEEE Conf. Computer Communications (INFOCOM), vol. 2, 2000, pp. 775–784.

[24] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 37, no. 6, pp. 1067–1080, Nov 2007.

[25] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Ef cient privacy-preserving face recognition," in Intl. Conf. Information Security and Cryptology (ICISC). Springer, 2010, pp. 229–244.

[26] T. Seidl and H.-P. Kriegel, "Optimal multi-step k-nearest neighbor search," SIGMOD Rec., vol. 27, no. 2, pp. 154–165, Jun. 1998.

[27] I. Damgard, M. Jurik, and J. B. Nielsen, "A generalization of Paillier's public-key system with applications to electronic voting," International Journal of Information Security, vol. 9, no. 6, pp. 371–385, Dec 2010.

[28] I. Damgard, M. Geisler, and M. Krøigaard, "Ef cient and secure comparison for on-line auctions," in Australasian Conf. Information Security and Privacy (ACISP). Springer, 2007, pp. 416–430.

[29] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally ef cient multi-authority election scheme," European transactions on Telecommunications, vol. 8, no. 5, pp. 481–490, 1997.

[30] J. Dossogne and F. La tte, "Blinded additively homomorphic encryption schemes for self-tallying voting," Journal of Information Security and Applications, vol. 22, pp. 40 – 53, 2015.

[31] C. E. Shannon, "Communication theory of secrecy systems," The Bell System Technical Journal, vol. 28, no. 4, pp. 656–715, Oct 1949.

[32] A. Yao, "How to generate and exchange secrets," in IEEE Symp. Foundations of Computer Science (FOCS), 1986.

[33] T. Schneider, Engineering Secure Two-Party Computation Protocols: Design, Optimization, and Applications of Ef cient Secure Function Evaluation. Springer, 2012.

[34] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, "Ef cient garbling from a xed-key blockcipher," in IEEE Symposium on Security and Privacy (IEEE S&P), May 2013, pp. 478–492.

[35] S. R. Tate and K. Xu, "On garbled circuits and constant round secure function evaluation," University of North Texas, Tech. Rep., 2003.

[36] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," in Intl. Colloquium on Automata, Languages and Programming (ICALP). Springer, 2008, pp. 486–498.

[37] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams, "Secure two-party computation is practical," in Advances in Cryptology — ASIACRYPT. Springer, 2009, pp. 250–267.

[38] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole," in Advances in Cryptology — EUROCRYPT. Springer, 2015, pp. 220–250.

[39] D. Demmler, T. Schneider, and M. Zohner, "ABY - a framework for ef cient mixed-protocol secure two-party computation," in Network and Distributed System Security Symposium (NDSS). 2015.

[40] Z. Yang and K. Järvinen, "Modeling privacy in WiFi ngerprinting indoor localization," in ProvSec. Springer, 2018, pp. 329–346.

[41] V. Kolesnikov and R. Kumaresan, "Improved OT extension for transferring short secrets," in Advances in Cryptology–CRYPTO. Springer, 2013, pp. 54–70.

[42] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More ef cient oblivious transfer and extensions for faster secure computation," in ACM SIGSAC Conf. Computer & Communications Security (CCS). ACM, 2013, pp. 535–548.

[43] S. A. Cook and S. O. Aanderaa, "On the minimum computation time of functions," Transactions of the American Mathematical Society, vol. 142, pp. 291–314, 1969.

[44] E. S. Lohan, J. Torres-Sospedra, H. Leppäkoski, P. Richter, Z. Peng, and J. Huerta, "Wi-Fi crowdsourced ngerprinting dataset for indoor positioning," Data, vol. 2, no. 4, 2017.

[45] C. Koç, "Analysis of sliding window techniques for exponentiation," Computers & Mathematics with Applications, vol. 30, no. 10, pp. 17 – 24, 1995.

[46] P. L. Montgomery, "Modular multiplication without trial division," Mathematics of Computation, vol. 44, no. 170, pp. 519–521, 1985.

[47] P. C. Kocher, "Timing attacks on implementations of Dif e-Hellman, RSA, DSS, and other systems," in Advances in Cryptology—CRYPTO, 1996, pp. 104–113.

[48] P. Richter, Z. Yang, O. Tkachenko, H. Leppäkoski, K. Järvinen, T. Schneider, and E. S. Lohan, "Received signal strength quantization for secure indoor positioning via ngerprinting," in Intl. Conf. Localization and GNSS (ICL-GNSS), 2018.

[49] E. B. Barker, W. C. Barker, W. E. Burr, W. T. Polk, and M. E. Smid, "SP 800-57. Recommendation for key management, part 1: General (revised)," NIST, Tech. Rep., 2007.

[50] M. Abdalla, T. E. Bjørstad, C. Cid, B. Gierlichs, A. H ülsing, A. Luykx, K. G. Paterson, B. Preneel, A.-R. Sadeghi, T. Spies, M. Stam, M. Ward, B. Warinschi, and G. Watson, "Algorithms, key size and protocols report," ECRYPT - CSA, Tech. Rep., 2018.

[51] S. D. Galbraith, "Elliptic curve Paillier schemes," Journal of Cryptology, vol. 15, no. 2, pp. 129–138, 2002.

[52] R. Agrawal and R. Srikant, Privacy-preserving data mining. ACM, 2000, vol. 29, no. 2.

[53] Y. Lindell and B. Pinkas, "Privacy preserving data mining," Journal of Cryptology, vol. 15, no. 3, pp. 177—-206, 2002.

[54] C. C. Aggarwal and P. S. Yu, A General Survey of Privacy-Preserving Data Mining Models and Algorithms. Springer, 2008, pp. 11–52.

Raine Nieminen received the M.Sc. (Tech.) degree in computer, communication and information sciences from Aalto University in Finland in 2018. He had short-term Research and Teaching Assistant positions in Aalto University in 2016 and 2017. In 2018, he was a full-time Research Assistant with the Department of Computer Science in University of Helsinki in Finland. From 2019 to 2020, he was a Security Specialist in Insta Digital in Finland. His research interests lie in the domains of security and cryptography.

Kimmo J ärvinen received the M.Sc. (Tech.) and D.Sc. (Tech.) degrees in electrical engineering from Helsinki University of Technology (TKK) in Finland in 2003 and 2008, respectively. From 2008 to 2013 and from 2015 to 2016, he was a postdoctoral researcher in the Department of (Information and) Computer Science in Aalto University in Finland. From 2014 to 2015, he was with the COSIC Group in KU Leuven ESAT in Belgium. Since 2016, he has been a Senior Researcher with the Department of Computer Science in University of Helsinki in Finland. His research interests are in the domains of security, cryptography, and cryptographic engineering.